FPT Education

FPT POLYTECHNIC

LẬP TRÌNH ANDROID 2

RECYCLERVIEW

www.poly.edu.vn

GIỚI THIỆU BÀI HỌC

- Tổng quan RecyclerView
- So sánh giữa RecyclerView và ListView
- Sử dụng RecyclerView

Mục tiêu

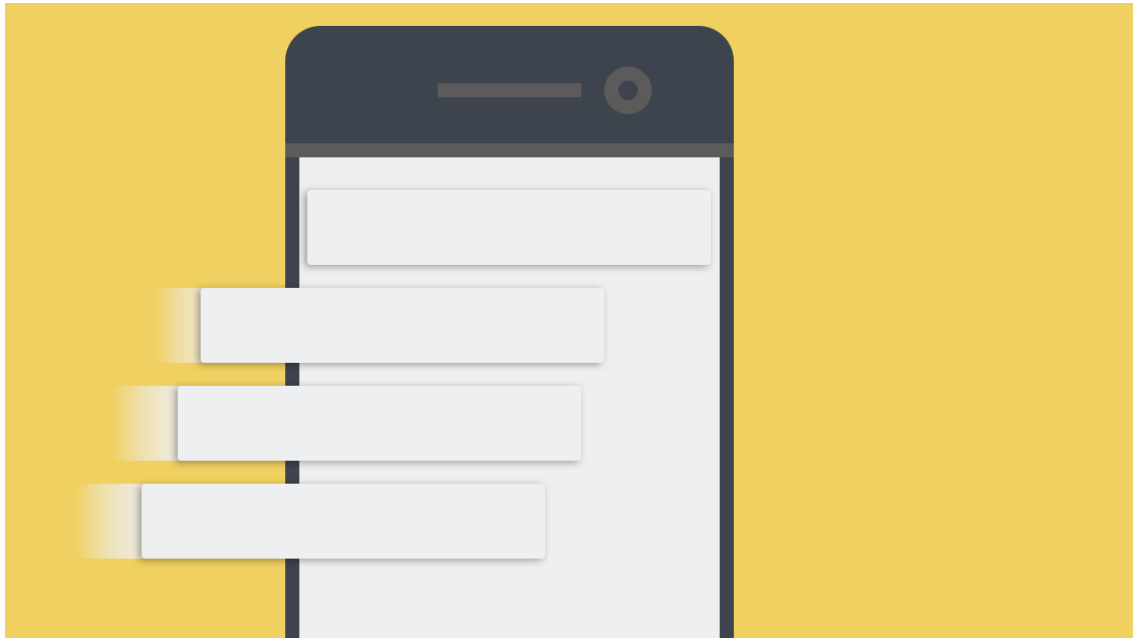
- Tổng quan RecyclerView
- So sánh giữa RecyclerView và ListView
- Sử dụng RecyclerView

GIỚI THIỆU VỀ RECYLERVIEW

...

TỔNG QUAN VỀ RECYCLERVIEW

- **RecyclerView** nó dùng để xây dựng UI gần giống với hoạt động của ListView, GridView.
- Nó biểu diễn danh sách với nhiều cách trình bày khác nhau, theo chiều đứng, chiều ngang.
- Nó là thư viện hỗ trợ tốt hơn ListView rất nhiều, nhất ra sử dụng trong *CoordinatorLayout* để tương tác với các thành phần UI khác.



LÀM VIỆC VỚI RECYCLERVIEW

- Khi dùng đến RecylerView thì bạn cũng cần làm việc với:
  - **RecyclerView.Adapter** Quản lý dữ liệu và cập nhật dữ liệu cần hiện thị vào View (phần tử hiện thị trong RecyclerView)
  - **RecyclerView.LayoutManager** Lớp mà để quy định cách mà vị trí các phần tử trong RecyclerView hiện thị, có thể sử dụng các lớp kế thừa LinearLayoutManager, GridLayoutManager
  - **RecyclerView.ItemAnimator** Lớp để xây dựng cách thực hoạt hình (động) cho các sự kiện trên phần tử hiện thị, như hiệu ứng khi thêm phần tử vào, xóa phần tử khỏi RecyclerView
  - **RecyclerView.Viewholder** lớp dùng để gán / cập nhật dữ liệu vào các phần tử.

SỬ DỤNG ADAPTER THAY ĐỔI PHẦN TỬ TRONG RECYCLERVIEW

- Để thêm bớt phần tử trong **RecyclerView** là phải thông qua Adapter của nó. Các phương thức Adapter thông báo đến RecylerView có thể sử dụng như:

Phương thức	Sử dụng
notifyItemChanged(int pos)	Cho biết phần tử ở vị trí pos thay đổi.
notifyItemInserted(int pos)	Thông báo Phần tử ở vị trí pos mới thêm vào
notifyItemRemoved(int pos)	Thông báo Phần tử ỏ vị trí pos bị loại bỏ
notifyDataSetChanged()	Thông báo toàn bộ dữ liệu thay đổi

- Trong **RecyclerView** có một số phương thức xử lý cuộn có thể sử dụng:

Phương thức	Áp dụng
scrollToPosition(int position)	Cuộn lập tức đến phần tử position
smoothScrollToPosition(int position)	Cuộn đến phần tử position (trôi đến phần tử)
scrollBy(int x, int y), smoothScrollBy(int dx, int dy)	Cuốn từ trạng thái hiện tại thêm một đoạn x, y theo phương dọc và ngang (lưu ý ảnh có tác động theo chiều X, Y không còn phụ thuộc loại LayoutManager) trình bày sau
scrollTo(int x, int y)	Cuộn đến tọa độ x, y

CUỘN TRONG RECYCLERVIEW

- Trong **RecyclerView** có một số phương thức xử lý cuộn có thể sử dụng:
  - Ví dụ: Cuộn đến phần tử đầu tiên sau khi cập nhật

```
adapter.notifyItemChanged(0);
recyclerView.scrollToPosition(0);
```

  - Ví dụ: Cuồn đến vị trí của phần tử mới được thêm vào

```
adapter.notifyItemChanged(students.size() -1);
recyclerView.scrollToPosition(students.size() -1);
```

SỬ DỤNG LINEARLAYOUTMANAGER

- **LinearLayoutManager** - cung cấp hai loại các phần từ xếp theo hàng thẳng đứng hoặc nằm ngang

```
int orientation = LinearLayoutManager.HORIZONTAL; //Cuộn ngang
// int orientation = LinearLayoutManager.VERTICAL; //cuộn đứng
LinearLayoutManager layoutManager = new LinearLayoutManager(this,
                                        LinearLayoutManager.HORIZONTAL, false);
layoutManager.scrollToPosition(0); //Thiết lập phần tử mặc định nếu muốn
// Gắn vào RecylerView
recyclerView.setLayoutManager(layoutManager);
```

❑ Một số phương thức trong **LinearLayoutManager**

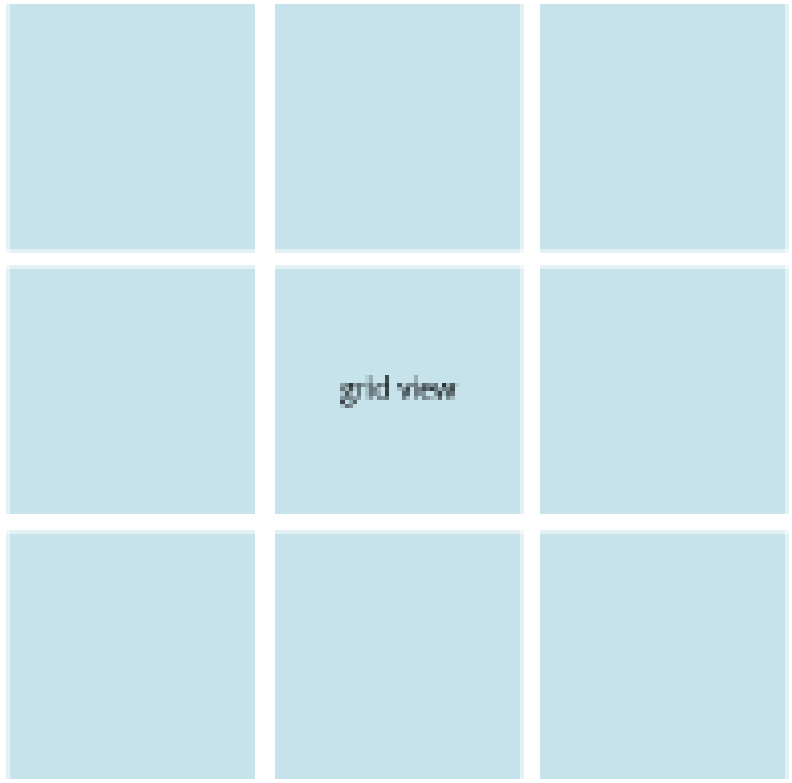
Phương thức	Áp dụng
findFirstCompletelyVisibleItemPosition() findLastCompletelyVisibleItemPosition()	Trả về vị trí của phần tử thứ nhất/cuối cùng đang xuất hiện trọn vẹn trong View, RecyclerView.NO_POSITION nếu không thấy
findFirstVisibleItemPosition() findLastVisibleItemPosition()	Trả về vị trí của phần tử thứ nhất/cuối cùng đang xuất hiện trong View, RecyclerView.NO_POSITION nếu không thấy
findViewByPosition(int position)	Trả về View trình diễn phần tử thứ position của Adapter, trả về null nếu phần tử đó chưa hiện thị trong View
scrollToPosition(int position)	Cuộn tới phần tử thứ position trong Adapter

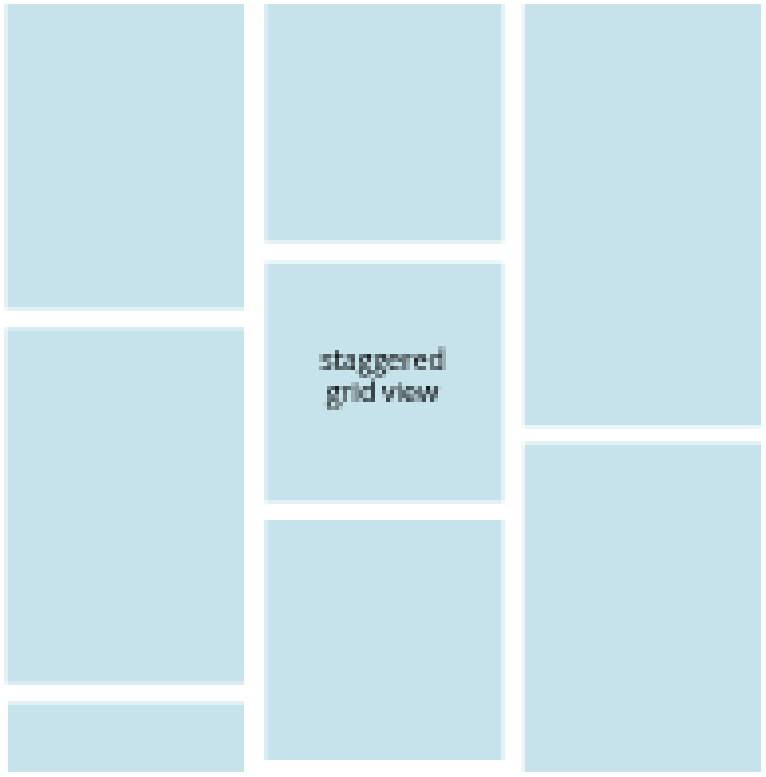
SỬ DỤNG GRIDLAYOUTMANAGER

- Có thể thiết lập RecyclerView hiển thị các phần tử ở dạng lưới. Có thể chọn lưới vuốt đứng, với số cột hiện thị cố định hoặc lưới vuốt ngang với số dòng cố định

```
int spanCount = 2; //Số cột nếu thiết lập lưới đứng, số dòng nếu lưới ngang
int orientation = GridLayoutManager.VERTICAL; //Lưới ngang
//int orientation = GridLayoutManager.HORIZONTAL; //Lưới dọc
GridLayoutManager gridLayoutManager = new GridLayoutManager(this, 2);
gridLayoutManager.setOrientation(GridLayoutManager.VERTICAL);
recyclerView.setLayoutManager(gridLayoutManager);
```

- Về **StaggeredGridLayoutManager** thì nó cũng là dạng lưới, và cách code giống với ***GridLayoutManager***, nhưng có một chút khác biệt khi hiển thị





SỬ DỤNG STAGGEREDGRIDLAYOUTMANAGER

- Về **StaggeredGridLayoutManager** thì nó cũng là dạng lưới, và cách code giống với ***GridLayoutManager***, nhưng có một chút khác biệt khi hiển thị
  - **Ví dụ:**

```
StaggeredGridLayoutManager gridLayoutManager = new StaggeredGridLayoutManager(2, StaggeredGridLayoutManager.VERTICAL);
recyclerView.setLayoutManager(gridLayoutManager);
```

SỬ DỤNG DIVIDERITEMDECORATION

- Đây là lớp kế thừa **RecyclerView.ItemDecoration** thư viện có sẵn để kẻ ngang hoặc đứng giữa các phần tử

- **Ví dụ:**

```
//Chèn một kẻ ngang giữa các phần tử
DividerItemDecoration dividerHorizontal = new DividerItemDecoration(this,
                                                    DividerItemDecoration.VERTICAL);
recyclerView.addItemDecoration(dividerHorizontal);

//Chèn một kẻ đứng giữa các phần tử
DividerItemDecoration dividerVertical = new DividerItemDecoration(this,
                                                    DividerItemDecoration.HORIZONTAL);
recyclerView.addItemDecoration(dividerVertical);
```

SỬ DỤNG DIVIDERITEMDECORATION

- **DividerItemDecoration** có phương thức **setDrawable** để bạn thiết lập ảnh riêng dùng để vẽ đường kẻ nếu muốn
- **Ví dụ:** Tạo ra một XML Drawable

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
android:shape="rectangle">
        <size android:width="1000px"
              android:height="30px" />
        <gradient android:startColor="#000000"
              android:centerColor="#beacac"
              android:endColor="#e63232"
              android:angle="0" />
</shape>
```

SỬ DỤNG DIVIDERITEMDECORATION

- **DividerItemDecoration** có phương thức **setDrawable** để bạn thiết lập ảnh riêng dùng để vẽ đường kẻ nếu muốn
- **Ví dụ:** Sử dụng để vẽ đường ngang

```
//Chèn một kẻ ngang giữa các phần tử
DividerItemDecoration dividerHorizontal = new
                    DividerItemDecoration(this, DividerItemDecoration.VERTICAL);
dividerHorizontal.setDrawable(ContextCompat.getDrawable(this,
R.drawable.devider_red)); recyclerView.addItemDecoration(dividerHorizontal);
```

SỬ DỤNG RECYLERVIEW

...

- **Ví dụ:** Hiển thị danh sách sản phẩm bằng RecyclerView
  - **Bước 1:** Định lớp **SanPham**

```
public class SanPham {
    3 usages
    private int masp;
    3 usages
    private String tensp;
    3 usages
    private int gia;

    1 usage
    public SanPham(int masp, String tensp, int gia) {
        this.masp = masp;
        this.tensp = tensp;
        this.gia = gia;
    }
    1 usage
    public int getMasp() { return masp; }

    public void setMasp(int masp) { this.masp = masp; }

    3 usages
    public String getTensp() { return tensp; }

    public void setTensp(String tensp) { this.tensp = tensp; }

    3 usages
    public int getGia() { return gia; }

    public void setGia(int gia) { this.gia = gia; }
}
```

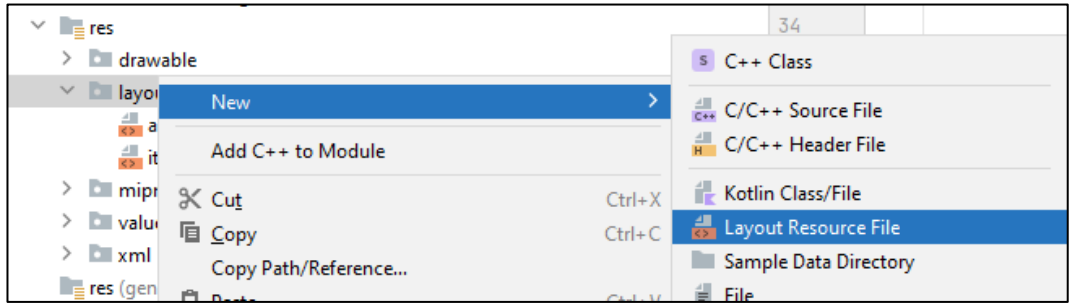
- **Ví dụ:** Hiển thị danh sách sản phẩm bằng RecyclerView
  - **Bước 2:** Tạo recyclerview trong layout

```
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recyclerDanhSach"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

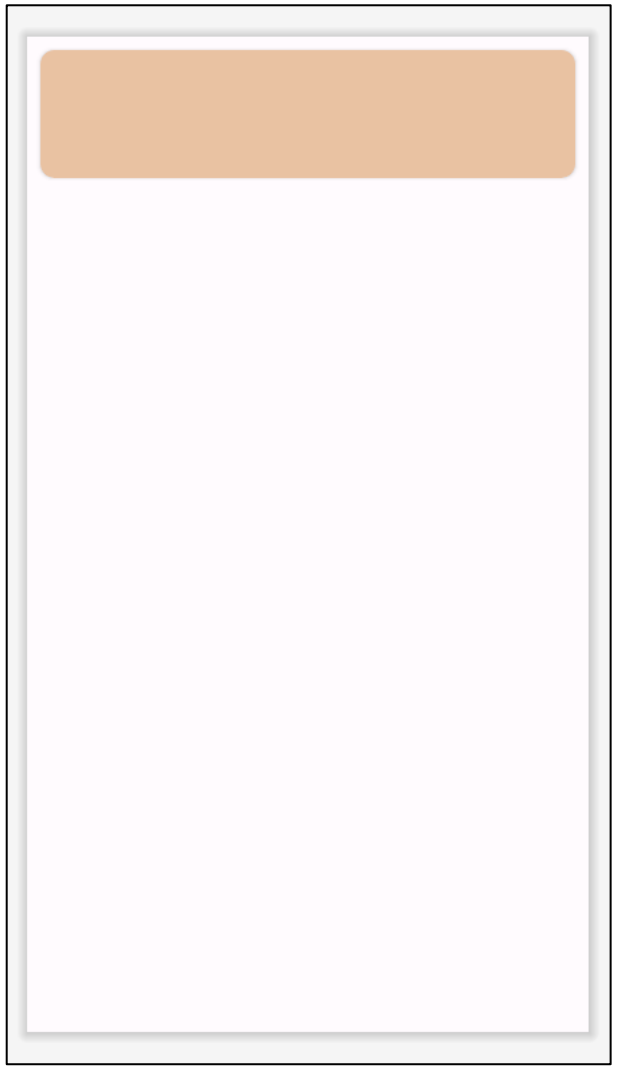
- **Ví dụ:** Hiển thị danh sách sản phẩm bằng RecyclerView
  - **Bước 2:** Tạo recyclerview trong layout

```
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recyclerDanhSach"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

- **Ví dụ:** Hiển thị danh sách sản phẩm bằng RecyclerView
  - **Bước 2:** Tạo layout cho item hiển thị trên recyclerview




```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_margin="10dp"
    app:cardCornerRadius="10dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#F1C09E"
        android:orientation="vertical">

        <TextView
            android:id="@+id/txtTenSP"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="40sp" />

        <TextView
            android:id="@+id/txtGia"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="30sp" />
    </LinearLayout>

</androidx.cardview.widget.CardView>
```



❑ **Ví dụ:** Hiển thị danh sách sản phẩm bằng RecyclerView

Bước 3: Tạo **SanPhamAdapter** extends **RecyclerView.Adapter** và **ViewHolder** để render item

- ❖ Tạo lớp **SanPhamAdapter** kế thừa **RecyclerView.Adapter**
- ❖ Định nghĩa thành phần **ViewHolder** (tương ứng **item_san_pham)**
- ❖ Ánh xạ **view** tương ứng
- ❖ Tạo danh sách ArrayList<SanPham> để chuẩn bị đưa dữ liệu vào Adapter
- ❖ Mỗi Adapter có 3 thành phần chính:

onCreateViewHolder, onBindViewHolder, getItemCount

- **Ví dụ:** Hiển thị danh sách sản phẩm bằng RecyclerView

```
public class SanPhamAdapter extends RecyclerView.Adapter<SanPhamAdapter.ViewHolder> {
    1 usage
    private final Context context;
    1 usage
    private final ArrayList<SanPham> list;

    1 usage
    public SanPhamAdapter(Context context, ArrayList<SanPham> list) {
        this.context = context;
        this.list = list;
    }

    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) { return null; }

    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {}

    @Override
    public int getItemCount() { return 0; }

    3 usages
    public static class ViewHolder extends RecyclerView.ViewHolder{
        1 usage
        TextView txtTenSP, txtGiaSP;
        public ViewHolder(@NonNull View itemView) {
            super(itemView);
            txtGiaSP = itemView.findViewById(R.id.txtGia);
            txtTenSP = itemView.findViewById(R.id.txtTenSP);
        }
    }
```

- **Ví dụ:** Hiển thị danh sách sản phẩm bằng RecyclerView

Bước 4: Implements các thành phần chính trong **SanPhamAdapter**

- **onCreateViewHolder**: cần chuyển layout ***item_san_pham.xml*** sang view sử dụng LayoutInflater
- **OnBindViewHolder**: Render **item_san_pham** ở vị trí position
- **getItemCount**: Số lượng item hiên thị ở recyclerView

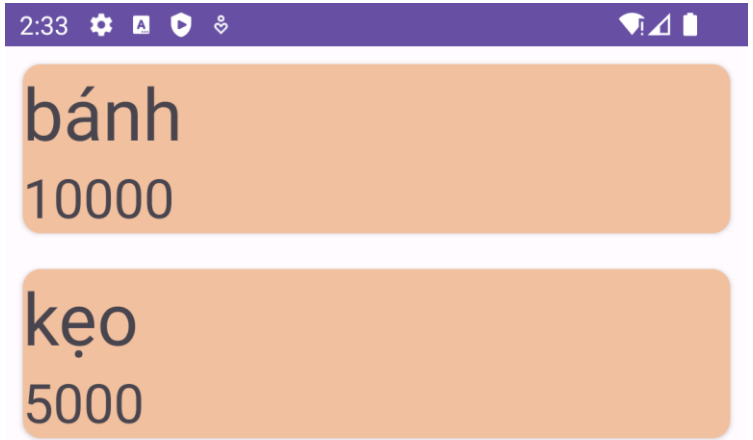
- **Ví dụ:** Hiển thị danh sách sản phẩm bằng RecyclerView

```
@NonNull
@Override
public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    LayoutInflater inflater = ((Activity)context).getLayoutInflater();
    View view = inflater.inflate(R.layout.item_san_pham, parent,  attachToRoot: false);
    return new ViewHolder(view);
}

@Override
public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
    holder.txtTenSP.setText(list.get(position).getTensp());
    holder.txtGiaSP.setText(String.valueOf(list.get(position).getGia()));
}

@Override
public int getItemCount() {
    return list.size();
}
```

- **Ví dụ:** Hiển thị danh sách sản phẩm bằng RecyclerView

Bước 5: Sử dụng trong Activity & Kết quả

```
SanPhamDAO sanPhamDAO = new SanPhamDAO( context: MainActivity.this);
ArrayList<SanPham> list = sanPhamDAO.getDS();
LinearLayoutManager linearLayoutManager = new LinearLayoutManager( context: MainActivity.this);
recyclerDanhSach.setLayoutManager(linearLayoutManager);
SanPhamAdapter adapter = new SanPhamAdapter( context: MainActivity.this,list);
recyclerDanhSach.setAdapter(adapter);
```



KHÁC BIỆT SO VỚI LISTVIEW

- Có thể fixed Size để tăng performance (mặc định scrollable)
  - recyclerViewSanPham.setHasFixedSize(true);
- Scroll ở vị trí bất kì
  - recyclerViewSanPham.scrollToPosition(2);
- Dễ dàng điều chỉnh layout:

```
LinearLayoutManager layoutManager = new LinearLayoutManager(this,
                                        LinearLayoutManager.HORIZONTAL, false);
layoutManager.scrollToPosition(1);
recyclerViewSanPham.setLayoutManager(layoutManager);
```

KHÁC BIỆT SO VỚI LISTVIEW

- Custom việc chia giữa các mục trong danh sách (**DividerItemDecoration**)

```
RecyclerView.ItemDecoration itemDecoration = new DividerItemDecoration(this,
                                                    DividerItemDecoration.VERTICAL);
recyclerViewSanPham.addItemDecoration(itemDecoration);
```

- Animators
  - 1. Cài đặt thêm thư viện ở Grade

```
implementation 'jp.wasabeef:recyclerview-animators:4.0.2`
```

  - 2. Sử dụng

```
recyclerViewMonHoc.setItemAnimator(new SlideInUpAnimator());
```

Touch Event: **addOnItemTouchListener**

```
recyclerDanhSach.addOnItemTouchListener(new RecyclerView.OnItemTouchListener() {
    @Override
    public boolean onInterceptTouchEvent(@NonNull RecyclerView rv, @NonNull MotionEvent e) {
        return false;
    }

    @Override
    public void onTouchEvent(@NonNull RecyclerView rv, @NonNull MotionEvent e) {

    }

    @Override
    public void onRequestDisallowInterceptTouchEvent(boolean disallowIntercept) {

    }
});
```


FPT Education
FPT POLYTECHNIC
Thank you
TIN HỌC VĂN PHÒNG QUỐC TẾ THEO CHUẨN ICDL
Syllabus 5 - Office 2007
LUẬT KINH TẾ
KỸ THUẬT SOẠN THẢO VÀ TRÌNH BÀY VĂN BẢN